Grid Computing: The Trend Of The Millenium

Ihssan Alkadi, (E-mail: ialkadi@louisiana.edu), University of Louisiana at Lafayette
Sandeep Gregory, University of Louisiana at Lafayette

ABSTRACT

A grid can be simply defined as a combination of different components which function collectively as a part of one large electrical or electronic circuit. The term "Grid Computing" can similarly be applied to a large number of computers which connect together to collectively solve a problem (which may be of scientific interest in most cases) of very high complexity and magnitude. The fundamental idea behind the making of any computer based grid is to utilize the idle time of processor cycles. Simply stated, a processor during the times it would stay idle would now team up with similar idle processors to tackle various complexities. The role each processor plays is very carefully defined and there is utmost transparency in the working of each processor/computer in a grid. This is called the "Division of Labor" in the smart world of intelligent computing. In lay man terms this is equivalent to a student and his group of friends collectively solving a single assignment which contains more than a single problem. The solution is trivial but the effort is collective. A grid computing environment may take many forms. It could be molded as a cluster based, distributed computing environment based or peer-to-peer system. A cluster based environment would see a central computer often called a "cluster head" distributing or maintaining a job schedule of the other computers in the grid. A distributed environment is seen often in the web environment. For example when a user requests a popular web page from the web server and if the web server is experiencing traffic congestion, then the user is re-routed to the same page on a different web server. The transition takes place so rapidly that the momentary delay due to server bottleneck problems is hardly felt. Peer to peer computing can be best explained through music download engines. If a user has a file he decides to share it via the web, other users needing the same file copy it through their music download engines. With great computing power comes great responsibility. Security is of utmost importance in a grid environment. Since a grid performs large computations, data is assumed to be available at every node in the processing cycle. This increases the risk of data manipulation in various forms. Also we have to keep in mind what happens to the data when a node fails. An ideal grid will have a small time of convergence and a low recovery time in case of a complete grid failure. By convergence, we mean that each and every processor node will have complete information about each and every other processor node in the grid. Recovery time is the time it takes for the grid to start from scratch after a major breakdown. To put things in the right perspective, a good grid based computing environment will have an intelligent grid administrator to monitor user logs and scheduled jobs and a good grid operating system which will be tailor made to suit the application of the grid. We propose a similarity between the OSI model and a Grid Model to elaborate the functions and utilities of a grid. We also try to propose a queuing theory for Grid Computing. There have been numerous previous comparisons and each is knowledgeable in its own right. But a similarity with a network model adds more weight since physically a grid is nothing but an interconnection, and interconnection can be best defined in relation to a computer network interconnection. How do users access networked computers, how are files shared and what are the levels of security are best explained through a networked computer system.

THE NATURE OF GRID ARCHITECTURE

This paper tries to identify key components, the nature of these components and to identify the interaction of these components in a grid based environment. The main concern is the interoperability between components. Just as the web provides common syntaxes and protocols for information sharing, we need similar protocols to facilitate interoperability in a grid. Protocols define how different system components will interact with each other to achieve a specified goal. A service is a language that a protocol speaks and the implementation of its behavior [2]. We now try and relate the grid layers to the OSI model. The OSI model

has seven different layers which are divided into two groups. The top three layers define interoperability between end points, whereas the bottom four layers define the end to end data transmission. The top layers know nothing about the bottom layers. They just perform operations assigned to them and the same is the case with the bottom layers. This interoperability between layers without knowledge of neighboring layers makes the OSI model a successful one and as such even grids should be and in most cases are implemented in such a manner.

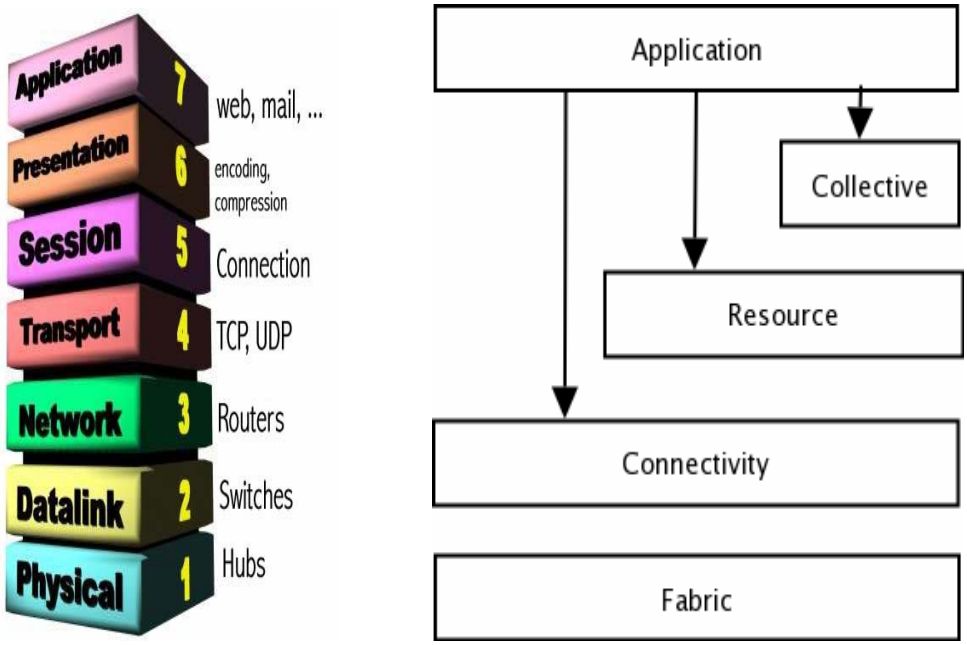

Fig 1. Relation Between The OSI Model And The Grid Layers Model

The Application Layer in both the OSI model and the Grid Layer model perform nearly identical functions. In short, the Application Layer is where users communicate to the computer. The Application Layer is responsible for identifying and establishing the availability of the intended communication partners and if sufficient resources are available for the intended operation. The Presentation Layer of the OSI model gets its name from its purpose. It provides data to the Application Layer. Basically, a Presentation Layer acts as a translator of resources and provides a host of conversion and coding functions. The Collective Layer in the Grid Layer maps to the Presentation Layer in the sense that it contains the necessary protocols and services necessary to link the lower layers to the Application Layer. It also stores the results of interactions between the various lower layers.

The Session Layer is responsible for setting up, managing and terminating sessions between Presentation Layer entities. This layer provides dialog control and coordinates communication between different systems. Similar in function to the Session layer, the Resource Layer uses protocols and security functions of the Connectivity Layer to initiate, secure, monitor, control and terminate operations on individual resources in a grid. The Transport Layer provides end to end data connectivity in the OSI model by establishing a logical connection between the sender and the receiver. The Network Layer is responsible for transmitting data between devices which are not locally connected. The Connectivity Layer maps to a combination of the Transport and Network Layer in the OSI model. The Connectivity Layer contains all the necessary protocols and authentication services required to perform grid specific transactions. So even if a grid spans a large geographical area or on the contrary, is a locally situated grid, all the necessary commands are issued by the Connectivity Layer to ensure smooth functioning of the Grid. In short, the Connectivity Layer looks after the network transactions of a grid which maybe an intra-grid or an inter-grid.

The Data Link Layer and the Physical Layer of the OSI model combine together and can be mapped to the Fabric Layer of a grid based environment. The Data Link Layer ensures proper message delivery and breaks up messages from the Network Layer into code which is understandable by the Physical Layer. The Physical Layer is responsible for communication between various types of media. The actual communication taking place in an OSI model is through the Physical Layer. Similarly, the Fabric Layer is the lowermost layer in a grid environment. It contains all the resources which need to be shared by the grid. Resources may take the form of computational power, data storage or sensing activity by the grid. Data frames could be encoded as IP packets and shipped across WAN's to geographically remote sites. They can then be decoded and passed to the local channel, which will make two different networks appear as one large network. So just as the Physical Layer is where all the communication is taking place in the OSI model, the Fabric Layer is the central hub of grid activity in grid computing. In short, the Fabric Layer is similar to the plug of an electrical network. The plug connects to the socket and the Fabric Layer is the last step before access is granted so that grid utilization can begin.

Grid Computing And The Queuing Theory

In the early 1990's the Internet was a huge success. Looking at the success of the interconnection technology, an environment was conceived and proposed. This computational environment would include a large number of heterogeneous and dynamic resources. This heterogeneous platform later came to be known as a Grid Computing environment. With Grid Computing emerging as an important tool which aided in problem solving, which was often complex, it was becoming increasingly important to find means by which the problem can be measured, analyzed and the output predicted to a fair degree before it is applied to various levels of grid processing. The key to this approach was to define a Queuing Model or a job scheduling scheme which will make the task of the various components in the grid much easy and thus reduce the idle time of the entire grid unit. It would also reduce the dependencies which could appear in the picture at a later or advanced stage of computational processing. We summarize the queuing model or a job scheduling scheme which is mostly utilized in a Grid Computing Environment.

Between the Application Layer and the Collective Layer, requests for resource allocations are made. A task manager which is normally present at the Application layer is given the responsibility of finding the available resources. The task manager sends requests to a schedule manager which is present at the collective layer just to handle these requests. The schedule manager then looks into a central directory to check for available resources. Between the Collective Layer and the Resource Layer, the schedule manager executes various schedule based algorithms and identifies the best of the available resources which will give the most efficient results. Between the Resource Layer and the Connectivity Layer there is often a predictor mechanism which measures the performance of the system, and based on the current output collects evaluation results. These results are often used to estimate the rate of task completion. Between the Connectivity Layer and the Fabric Layer various sorts of data managers and performance measure tools are present. These tools measure system and application information utilizing sensors of various forms. Thus the presence of various managers or tools present at different levels of processing ensure that once the problem is input to the system, the problem would logically pass through a series of operations which would involve a queuing based approach to reach an end result. Without a queuing based scheme there is always a possibility that the available resources are not being utilized in an efficient manner.

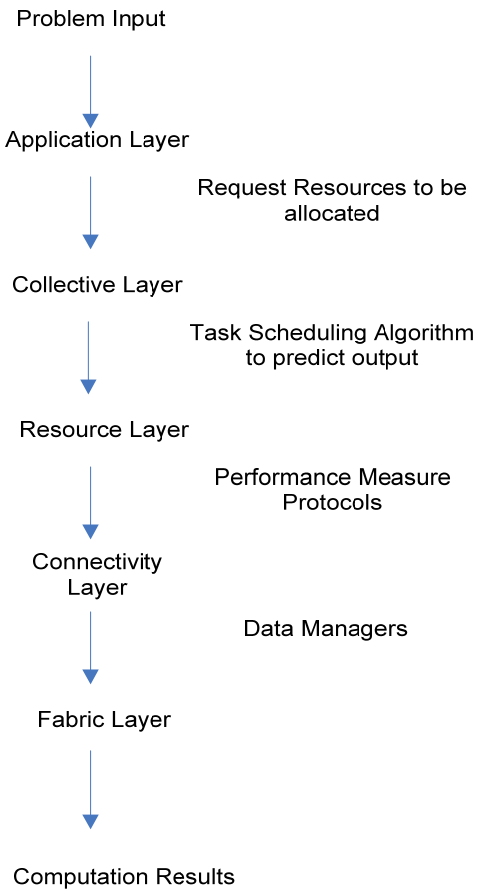

Fig 2 Integration Of The Grid Computing Environment And A Job Based Scheduling Scheme

BUILDING A SECURE GRID

As the scalability of a grid increases, the problems associated with the smooth functioning of a grid tend to grow proportionately. The main areas of concern are security, resource allocation, scheduling tasks, data access (which may not be available locally), policy management, fault tolerance, failure recovery, site autonomy and quality of service (QOS) [3].

Security

Security is modern grids is needed in the form of secure login's, authentication and authorization, access rights and privileges. Data encryption is the need of the hour to prevent intrusion on data transfers, data interceptions, data tampering and network disruptions. Elliptic Curve Cryptography, which is a form of Public Key Cryptography, which utilizes the algebraic properties of elliptic curves over finite fields, finds large use in processor based data encryptions and as such should be tried out on grid entities as well. A grid should be designed keeping the technical expertise of the enemy in mind. Remote window logins will become increasingly popular in the days to come where users will be able to access the grid entities without in the vicinity of their surroundings.

Resource Allocation

Resource Allocation in a grid can either be centralized or localized [4]. A centralized approach would not be ideally scalable i.e. the central point will soon become a bottleneck to the entire grid and this point could be seen as a central point of failure for the full grid based system. Similarly, a localized approach would tend to make applications which are aware of the grid infrastructure compete for available resources. Thus the grid administrator or the grid software should reasonably be aware of how much resources should be available to which application.

Scheduling Tasks

Time slots to perform computational tasks must be designed keeping in mind the user preferences, number of working hours, availability of machine resources and maintenance periods [5]. Proper synchronization is needed to minimize the idle time of computing machinery to generate maximum efficiency.

Data Access

At present data is distributed with the jobs to be performed by each application in the grid. This sometime involves delays due to dependencies of applications on one another. A simple example would be a two step calculation in which the second step requires problem solutions of the first step. So as long as the first step is being performed, the processors which will perform the second step remain idle. Grids of tomorrow should be designed keeping in mind data dependencies and this can be truly realized with a proper backbone of high speed communication networks which support fast data transfer capabilities.

Fault Tolerance

A major challenge for grids is facing faults and recovering from them [6]. Ideally data should be backed up after important computations or certain nodes should be administered data backing up capabilities. Another scheme would be storing computed data as image files which would be available locally so that a system can recover from scratch after a system crash.

Site Autonomy

Ideally a grid should incorporate Site Autonomy. By Site Autonomy, we mean that if a grid utilizes multiple servers, then each server should be administered independently with respect to the other servers in the grid. This would create a small non distributed environment in a large distributed environment. We could also incorporate local databases or directories to be used by each server, instead of one complete directory tree. By doing this, we are improving the scope to monitor, administer and install new software at key nodes in the grid. Also this gives administrators better control over the complete grid system.

Quality of Service (QoS)

Three parameters determine the QoS in a Grid Computing System. They are resilience, delay and system throughput. Grid systems should be designed to recover from fatal system crashes or alternatively protocols should be defined to balance the load on particular systems when network traffic tends to be on the higher side. To minimize the delay, proper job sequencing is a must to minimize the machine downtime. Alternatively, data access should be made available to applications which need numerous computing resources globally rather than locally. And finally, to improve system throughput the communication channel should be chosen after careful study and estimation. We cannot expect Tera Byte data transfer rates on a mismatched link.

CONCLUSION

If planned and designed correctly, grid based systems can solve a large number of relatively complex problems through inexpensive supercomputing. As time passes grid based systems will find increasing use in applications ranging from observing the output of embedded generators to supporting and managing customer relationships and supply chain management. However, the most traditional field where grid based systems still continues to be applied to, is Particle Physics. As the number of applications supported in a grid environment increase, concerns arising due to the economics, security, performance and grid standards will also increase. Simply put, just as it is unfeasible to perform routine tasks using a grid based system, similarly there is no point granting complete administrative access to novice users or users who do not require majority access to the software controlling the grid. Future work in grid based computing will involve not only scientific applications but also solutions to business problems, as it is now being proved that grid computing is a versatile tool which when used effectively can tackle almost any computational or business related problem from diverse fields. In this paper, we have presented an introduction to grid based computing, grid architecture and the factors which are responsible for a grid to function smoothly.

REFERENCES

1. James C Browne Grid Computing as Applied Distributed Computation: A Graduate Seminar on Internet and Grid Computing. IEEE International Symposium on Cluster Computing and the Grid (2004).
2. Ian Foster, Carl Kesselman and Steven Tuecke The Anatomy of the Grid – Enabling Scalable Virtual Organizations. Appeared in *International Supercomputer Applications* (2001).
3. Chao- Tung Yang and William C Chu *Grid Computing in Taiwan*. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04).
4. Teena Vyas *Grid Computing – Making the Global Infrastructure a reality* (March 11th 2004).
5. Marcin Akon, Marcin Lawenda, Norbert Mayer, Dominik Stoklosa, Tomasz Rajtar, Damian Kaliszan, Maciej Stroinski *Scheduling Interactive tasks in Grid Based Systems*. Poznan Supercomputing and Network Center.
6. Pawel Garbacki, Bartosz Biskupkski, Henri Bal Transparent Fault Tolerance for Grid Applications.
7. Viktor Berstis *Fundamentals of Grid Computing* Redpaper (REDP-3613-00). Published on 12th November 2002.
8. Fredrica Darema Grid Computing and Beyond: The Context of Dynamic Data Driven Applications. Proceedings of IEEE (March 2005).
9. Malcolm Irving, Gareth Taylor and Peter Hobson Plug into Grid Computing. *Power and Energy Magazine*, IEEE (March – April 2004).
10. Xian – He Sun, Ming Wu, Grid Harvest Service: A System for Long Term, Application – Level Task Scheduling.
11. Yan – Li Hu, Bao – Xin Xiu, Wei – Ming Zhang, Wel – Dong Xiao, Zhong Liu, An Algorithm for Jobs Scheduling in Computational Grid Based on Time – Balancing Strategy (Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guanzhou, 18 – 21 August 2005).